SYSTEMS AND METHODS FOR AUTHORING LIGHTING SEQUENCES

This application is based on, and claims the benefit of, U.S. Provisional Application No. 60/143,790, filed July 14, 1999.

Field of the Invention

The present invention relates generally to systems and methods for controlling lighting systems, and more particularly to computerized systems and methods for designing lighting sequences and executing such sequences on lighting systems.

Background of the Invention

Most modern-day lighting controllers are designed to control white light (or monochromatic light) in a theatrical or high-end business setting. A light producing monochromatic light, such as white, blue, or red, can be changed primarily along a single dimension – brightness – from off to a maximum brightness. Current controllers permit a user to specify a brightness for each light over time.

This method becomes increasingly more complicated for lights capable of changing the color of emitted light, because the resulting color and intensity is a combination of the intensity of three component primary colors, each of which can be set independent of the others for a particular light. Thus, the output is a function of three dimensions, rather than one, to be specified for each point in time, greatly increasing the effort and time involved in creating an effect. U.S. Patent No. 5,307,295 to Taylor et al. describes a system for creating lighting sequences which simplifies some aspects of creating a lighting sequence, but many of the parameters still need to be specified for each light, much as they would be on a standard lighting console. A more intuitive method for designing lighting sequences would not only simplify and speed up the designing process, but would permit users to design lighting sequences with less training and experience than is often necessary today.

Furthermore, although sequences can be created and played back by traditional methods, the content of the sequences typically progresses with time and is not subject to modification during playback. For example, if a dramatic scene requires a flash of lightning to be simulated at a certain time, this effect is typically achieved either by

meticulously timing the staging to make the programmed flash and the critical moment coincide, or by manually effecting the flash at the critical moment. Such techniques either require considerable reliance on chance or preclude reliance on automation.

A technique that permits an intuitive approach to designing lighting sequences would reduce the time and training required to achieve a desired effect, and would permit colored lights to be operated with a minimal impact on efficiency. Additionally, a method for executing such lighting sequences that promotes flexibility in the reproduction of the sequence will permit increased freedom in an associated performance, or allow use of programmed lighting sequences in situations which are inherently unpredictable.

Summary of the Invention

The systems and methods described herein relate to an intuitive interface for the design of lighting sequences, such as by providing a visual representation of a sequence as it is being designed. Additionally, the systems and methods described herein relate to reproduction of programmed lighting sequences such that the sequence can be modified during playback, e.g., based on external stimuli or cues.

A system for preparing a light sequence according to the principles of the invention may include an authoring interface displaying information representative of a plurality of lighting effects, and a sequence authoring module to permit a user to select a lighting effect, a lighting unit to execute the lighting effect, a start time for the lighting effect, and a stop time for the lighting effect.

A method for preparing a lighting sequence capable of being executed by a processor according to the principles of the invention may include providing a processor interface including information representative of a plurality of lighting effects, receiving information representative of a lighting unit, receiving information representative of a first lighting effect to be executed by the lighting unit, receiving information representative of a start time for the first lighting effect, and receiving information representative of a stop time for the first lighting effect.

In another aspect, a system for controlling a plurality of lighting units according to the principles of the invention may include a data interface for receiving instructions

for controlling a plurality of lighting units, a signal interface for receiving external signals, a processor for converting said instructions to a data stream and for altering the conversion of said instructions based on the external signals received, and a data output for transmitting the data stream to a plurality of lighting units.

In another aspect, a method for controlling a plurality of lighting units according to the principles of the invention may include receiving instructions for controlling a plurality of lighting units, receiving external signals, converting said instructions to a data stream based on the external signals received, and transmitting the data stream to a plurality of lighting units.

In another aspect, a method for controlling a plurality of lighting units according to the principles of the invention may include receiving instructions including a primary lighting effect and a secondary lighting effect, the secondary lighting effect designated to be executed instead of the primary lighting effect upon a predetermined condition, sending instructions to a lighting unit to execute the primary lighting effect, receiving a signal indicative of the predetermined condition, and sending instructions to the lighting unit to execute the secondary lighting effect.

In another aspect, a method for controlling a plurality of lighting units according to the invention may include receiving instructions for executing a timed sequence of lighting effects, executing the sequence of lighting effects utilizing a plurality of lighting units, receiving an external signal, and altering the execution of the sequence of lighting effects.

Brief Description of the Figures

The following figures depict certain illustrative embodiments of the invention in which like reference numerals refer to like elements. These depicted embodiments are to be understood as illustrative of the invention and not as limiting in any way.

Figure 1 illustrates a system for creating a lighting sequence and executing the lighting sequence on a plurality of lighting units as described herein.

Figure 2 presents an exemplary method for creating a lighting effect as described herein.

Figure 3 depicts a representative interface for describing an arrangement of lighting units.

Figure 4 represents an alternate interface for graphically reproducing a lighting sequence.

Figure 5 portrays a representative interface for creating a lighting sequence as described herein.

Figure 6 shows one embodiment of a controller for executing a lighting sequence as described herein.

Detailed Description of the Illustrated Embodiments

The description below pertains to several illustrative embodiments of the invention. Although many variations of the invention may be envisioned by one skilled in the art, such variations and improvements are intended to fall within the compass of this disclosure. Thus, the scope of the invention is not to be limited in any way by the disclosure below. The terms "sequence" or "light sequence", as used herein, are intended to refer to sequential displays, as well as non-sequential displays, flow-controlled displays, interrupt driven or event driven displays, or any other controlled, overlapping, or sequential displays with one or more lights.

The systems and methods described herein relate to a system, such as a processor 10 supporting a software application having an interface 15, as depicted in Figure 1, with which a user may create a lighting program 20, which may include one or more lighting sequences, capable of being executed by a lighting controller 30 which controls one or more lighting units 40. The term "sequence" in the context of this disclosure is used to refer to any pattern, show, sequence, arrangement or collection of commands used to operate lighting units or other devices through the system. One of skill in the art would recognize that a sequence would also not need to be an ordered sequence or have a linear design. Sequences comprising non-linear, priority-based, and/or overlapping commands may still comprise a sequence. The software application may be a stand-alone application, such as an executable image of a C++ or Fortran program or other

executable code and/or libraries, or may be implemented in conjunction with or accessible by a Web browser, e.g., as a Java applet or one or more HTML web pages, etc. Processor 10 may be any system for processing in response to a signal or data and should be understood to encompass microprocessors, microcontrollers, other integrated circuits, computer software, computer hardware, electrical circuits, application-specific integrated circuits, personal computers, chips, and other devices alone or in combination capable of providing processing functions. For example, processor 10 can be any suitable data processing platform, such as a conventional IBM PC workstation operating the Windows operating system, or a SUN workstation operating a version of the Unix operating system, such as Solaris, or any other suitable workstation. Controller 30 may communicate with lighting units 40 by radio frequency (RF), ultrasonic, auditory, infrared (IR), optical, microwave, laser, electromagnetic, or any other transmission or connection method or system. Any suitable protocol may be used for transmission, including pulse-width modulated signals such as DMX, RS-485, RS-232, or any other suitable protocol. Lighting units 40 may be incandescent, LED, fluorescent, halogen, laser, or any other type of light source, e.g., configured so that each lighting unit is associated with a predetermined assigned address either unique to that lighting unit or overlapping the address of other lighting units. In certain embodiments, a single component may be capable both of permitting a user to create a lighting program and controlling the lighting units, and the present invention is intended to encompass this and other variations on the system depicted in Figure 1 which can be used to implement the methods described below. In certain embodiments, the functions of the software application may be provided by a hardware device, such as a chip or card, or any other system capable of providing any of the functions described herein.

According to a method 200 for creating a lighting sequence set forth in Figure 2, a user may select from among a set of predetermined 'stock' effects 210. The stock effects function as discrete elements or building blocks useful for assembling a sequence. Additionally, a user may compose a particular sequence and include that sequence in the stock effects to eliminate the need for creating repeated elements *de novo* each time the effect is desired. For example, the set of stock effects may include a dimming effect and a brightening effect. A user may compose a pulse effect by specifying the alternation of the dimming and brightening effects, and include the pulse effect in the set of stock effects. Thus, each time a pulse effect is thereafter required, the

stock effect can be utilized without the need for repeatedly selecting dimming and brightening effects to achieve the same goal. In certain embodiments, stock effects may also be created by a user via any programming language, such as Java, C, C++, or any other suitable language. Effects may be added to the set of stock effects by providing the effects as plug-ins, by including the effects in an effects file, or by any other technique suitable for organizing effects in a manner that permits adding, deleting, and altering the set of effects.

Additionally, a user may select an effect and indicate a time at which that effect should begin 220. For example, the user may indicate that a brightening effect should start three minutes after a sequence commences. Additionally, the user may select an ending time or duration for the effect 230. Thus, by indicating that the effect should end five minutes after the sequence commences, or equivalently by indicating that the effect should last for two minutes, a user may set the time parameters of the brightening effect. Additional parameters may be specified by the user, as may be appropriate for the particular effect 240. For example, a brightening or dimming effect may be further defined by an initial brightness and an ending brightness. The rate of change may be predetermined, i.e., the dimming effect may apply a linear rate of dimming over the assigned timespan, or may be alterable by the user, e.g., may permit slow dimming at the beginning followed by a rapid drop-off, or by any other scheme the user specifies. Similarly, a pulse effect, as described above, might instead by characterized by a maximum brightness, a minimum brightness, and a periodicity, or rate of alternation. Additionally, the mode of alternation may be alterable by the user, e.g., the changes in brightness may reflect a sine function or alternating linear changes. In embodiments wherein color-changing lights are employed, parameters such as initial color, final color, rate of change, etc. may be specified by the user. Many additional effects and suitable parameters therefor are known or will be apparent to those of skill in the art, and fall within the scope of this disclosure.

In certain embodiments, a user may specify a transition between two effects which occur in sequence. For example, when a pulse effect is followed by a dimming effect, the pulse effect may alternate less rapidly, grow gradually dimmer, or vary less between maximum and minimum brightness towards the termination of the effect. Techniques for transitioning between these or other may be determined by the user for

each transition, e.g., by selecting a transition effect from a set of predetermined transition effects, or by setting transition parameters for the beginning and/or end of one or both effects.

In a further embodiment, users may specify multiple lighting effects for the same lighting unit that place effects overlapping in time or in location. These overlapping effects may be used in an additive or subtractive manner such that the multiple effects interact with each other. For example, a user could impose a brightening effect on a pulsing effect the brightening effect imposing the minimum brightness parameter of the pulse to give the effect of pulsing slowly growing to a steady light.

In another embodiment, the overlapping lighting effects could have priorities or cues attached to them which could allow a particular lighting unit to change effect on the receipt of a cue. This cue could be any type of cue, received externally or internally to the system, and includes, but is not limited to, a user-triggered cue such as a manual switch or bump button; a user-defined cue such as a certain keystroke combination or a timing key allowing a user to tap or pace for a certain effect; a cue generated by the system such as an internal clocking mechanism, an internal memory one, or a software based one; a mechanical cue generated from an analog or digital device attached to the system such as a clock, external light sensor, music synchronization device, sound level detection device, or a manual device such as a switch; a cue received over a transmission medium such as an electrical wire or cable, RF signal or IR signal; or a cue received from a lighting unit attached to the system. The priority could allow the system to choose a default priority effect that is the effect used by the lighting unit unless a particular cue is received, at which point the system instructs the use of a different effect. This change of effect could be temporary occurring only while the cue occurs or defined for a specified period, could be permanent not allowing for further receipt of other effects or cues, or could be priority based, waiting for a new cue to return to the original effect or select a new one. Alternatively, the system could select effects based on the state of a cue and the importance of a desired effect. For instance, if a sound sensor sensed sudden noise, it could trigger a high priority alarm lighting effect overriding all the effects otherwise present or awaiting execution. The priority could also be state dependent where a cue selects an alternative effect or is ignored depending on the current state of the system.

In certain embodiments, the outcome of one effect may be programmed to depend upon a second effect. For example, an effect assigned to one lighting unit may be a random color effect, and an effect assigned to a second lighting unit may be designated to match the color of the random color effect. Alternatively, one lighting unit may be programmed to execute an effect, such as a flashing effect, whenever a second lighting unit meets a certain condition, such as being turned off. Even more complex arrangements, such as an effect which is initiated upon a certain condition of one effect, matches the color of another effect, the rate of a third effect, can be created by this scheme. Other combinations of effects wherein at least one parameter or occurrence of an effect is dependent on a parameter or occurrence of a second effect will be apparent to those of skill in the art and are intended to fall within the scope of this disclosure.

In still other embodiments, the systems and methods described herein permit a lighting sequence to be influenced by external inputs during performance. For example, a lighting sequence or effect may be programmed to start upon receipt of a trigger signal, a sequence or effect may take precedence if a signal is received, a sequence or effect may be designated to repeat or continue until a signal is received, etc. Thus, instead of assigning a discrete start time to an effect or sequence, a user may instead designate that effect or sequence to begin when a certain stimulus is received. Furthermore, during creation, a user may designate two or more effects for overlapping or concurrent time periods and assign the effects different priorities or conditions to determine which effect is executed upon playback. In yet another embodiment, a user may link a parameter for an effect to an external input, including analog, digital and manual inputs, such that the color, speed, or other attribute of an effect may depend on a signal from an external device, measuring, for example, volume, brightness, temperature, pitch, inclination, wave length, or any other appropriate condition. Thus, the selection of a lighting sequence, the selection of an effect, or the selection of a parameter may be determined or influenced by input from an external source, such as a user, chronometer, device, or sensor.

In event-driven embodiments, such as those using external inputs and those using outputs of other effects as inputs, a menu may be provided to define inputs and the consequences thereof. For example, a palette of predetermined inputs may be provided to a user. Each input, such as a specified transducer or the output of another effect, may

be selected and placed within an authored lighting sequence as a trigger for a new effect, or as a trigger to a variation in an existing effect. Known inputs may include, for example, thermistors, clocks, keyboards, numeric keypads, Musical Instrument Digital Interface ("MIDI") inputs, DMX control signals, TTL or CMOS logical signals, other visual or audio signals, or any other protocol, standard, or other signaling or control technique having a predetermined form whether analog, digital, manual, or any other form. The palette may also include a custom input, represented as, for example, an icon in a palette, or an option in a drop-down menu. The custom input may allow a user to define the voltage, current, duration, and/or form (i.e., sinusoid, pulse, step, modulation) for an input signal that will operate as a control or trigger in a sequence.

For instance, a theatrical lighting sequence may include programmed lighting sequences and special effects in the order in which they occur, but requiring input at specified points before the next sequence or portion thereof is executed. In this way, scene changes may take place not automatically as a function of timing alone, but at the cue of a director, producer, stage hand, or other participant. Similarly, effects which need to be timed with an action on the stage, such as brightening when an actor lights a candle or flips a switch, dramatic flashes of lightning, etc., can be indicated precisely by a director, producer, stage hand, or other participant - even an actor - thereby reducing the difficulty and risk of relying on preprogrammed timing alone.

Input from sensors can also be used to modify lighting sequences. For example, a light sensor may be used to modify the brightness of the lights, for example, to maintain a constant lighting level regardless of the amount of sunlight entering a room, or to make sure a lighting effect is prominent despite the presence of other sources of light. A motion sensor or other detector may be used as a trigger to start or alter a lighting sequence. For example, a user may program a lighting sequence for advertising or display purposes to change when a person approaches a sales counter or display. Temperature sensors may also be used to provide input. For example, the color of light in a freezer may be programmed to be dependent on temperature, e.g., providing blue light to indicate cold temperature, changing gradually to red as the temperature rises, until a critical temperature is reached, whereupon a flashing or other warning effect may begin. Similarly, an alarm system may be used to provide a signal that triggers a lighting sequence or effect for providing a warning, distress signal, or other indication. An

interactive lighting sequence may be created, e.g., wherein the executed effect varies according to a person's position, movements, or other actions.

In certain embodiments, a user may provide information representative of the number and types of lighting units and the spatial relationships between them. For example, an interface 300 may be provided as depicted in Figure 3, such as a grid or other two-dimensional array, that permits the user to arrange icons or other representative elements to represent the arrangement of the lighting units being used. In one embodiment, depicted in Figure 3, the interface 300 provides to a user a selection of standard types of lighting units 310, e.g., cove lights, lamps, spotlights, etc., such as by providing a selection of types of lighting units in a menu, on a palette, on a toolbar, etc. The user may then select and arrange the lighting units on the interface, e.g., within layout space 320 in an arrangement which approximates the physical arrangement of the actual lighting units.

In certain embodiments, the lighting units may be organized into different groups, e.g., to facilitate manipulation of a large number of lighting units. Lighting units may be organized into groups based on spatial relationships, functional relationships, types of lighting units, or any other scheme desired by the user. Spatial arrangements can be helpful for entering and carrying out lighting effects easily. For example, if a group of lights are arranged in a row and this information is provided to the system, the system can then implement effects such as a rainbow or a sequential flash without need for a user to specify a separate and individual program for each lighting unit. All the above types of implementation or effects could be used on a group of units as well as on single lighting units. The use of groups can also allow a user to enter a single commend or cue to control a predetermined selection of lighting units.

A lighting sequence can be tested or executed on a lighting system to experience the effects created by the user. Additionally, the interface 300 may be capable of reproducing a lighting sequence created by the user, for example, by recreating the programmed effects as though the icons on the interface were the lighting units to be controlled. Thus, if a lighting sequence specified that a certain lighting unit gradually brightens to a medium intensity, upon playback, the icon representing that lighting unit may start black and gradually lighten to gray. Similarly, color changes, flashing, and other effects can be visually represented on the interface. This function may permit a

user to present a wholly or partially created lighting sequence on a monitor or other video terminal, pause playback, and modify the lighting sequence before resuming playback, to provide a highly interactive method for show creation. In a further embodiment, the system could allow fast-forwarding, reversing, rewinding, or other functions to allow editing of any portion of the lighting sequence. In a still further embodiment, the system could use additional interface features like those known in the art. This can include, but is not limited to, non-linear editing such as that used in the Adobe or such devices or controls as scrolls, drag bars, or other devices and controls.

An alternate interface 400 for reproducing a lighting sequence is presented in Figure 4. Interface 400 includes representations of lighting elements 410 and playback controls 420. Other techniques for visualizing a lighting sequence will be apparent to those of skill in the art and may be employed without departing from the scope and spirit of this disclosure.

An interface capable of representing the lighting sequence may also be used during entry of the lighting sequence. For example, a grid, such as interface 15 of Figure 1, may be employed, wherein available lighting units are represented along one axis and time is represented along a second axis. Thus, when a user specifies that a certain lighting unit gradually brightens to a medium intensity, the portion of the grid defined by that lighting unit, the start time, and the ending time may appear black at one end of the grid portion and gradually lighten to gray at the other end of the grid portion. In this way, the effect can be visually represented to the user on the interface as the lighting sequence is being created. In certain embodiments, effects that are difficult to represent with a static representation, such as flashing, random color changes, etc., can be represented kinetically on the interface, e.g., by flashing or randomly changing the color of the defined grid portion. An example of an interface 500 representing a sequence for an assortment of three lighting units are shown in Figure 5. Time chart 510 visually depicts the output of each of the three lights at each moment in time according to the temporal axis 515. At a glance, the user can readily determine what effect is assigned to any lighting unit at any point in time, simplifying the coordination of effects across multiple lighting units and allowing rapid review of the lighting sequence.

Additionally, Figure 5 depicts a palette 520 which includes the stock effects from which a user may select lighting effects, although other techniques for providing the set

of stock effects, such as by a menu, toolbar, etc., may be employed in the systems and methods described herein. In palette 520 there are provided icons for stock effects for the lighting of a fixed color effect 552, a cross fade between two color effects 554, a random color effect 558, a color high effect 560, a chasing rainbow effect 565, a strobe effect 564, and a sparkle effect 568. This list is by no means exhaustive and other types of effects could be included as would be obvious to one of skill in the art. To assign an effect to a lighting unit, the user may select an effect from the palette and select a region of the grid corresponding to the appropriate lighting unit or units and the desired time interval for the effect. Additional parameters may be set by any suitable technique, such as by entering numerical values, selecting options from a palette, menu, or toolbar, drawing a vector, or any other technique known in the art, such as the parameter entry field 525. Other interfaces and techniques for entry of lighting sequences suitable for performing some or all of the various functions described herein may be used and are intended to be encompassed by the scope of this disclosure. Examples of functions and interfaces suitable for use with the invention may be found in "A Digital Video Primer," June, 2000, by the Adobe Dynamic Media Group, Adobe Systems, Inc., incorporated herein by reference.

The methods described above can be readily adapted for controlling units other than lighting units. For example, in a theatrical setting, fog machines, sound effects, wind machines, curtains, bubble machines, projectors, stage practicals, stage elevators, pyrotechnical devices, backdrops, and any other features capable of being controlled by a computer may be controlled by a sequence as described herein. In this way, multiple events can be automated and timed. For example, the user may program the lights to begin to brighten as the curtain goes up, followed by the sound of a gunshot as the fog rolls over the stage. In a home, for example, a program can be used to turn on lights and sound an alarm at 7:00 and turn on a coffee maker fifteen minutes later. Holiday lighting arrays, e.g., on trees or houses, can be synchronized with the motion of mechanical figurines or musical recordings. An exhibit or amusement ride can coordinate precipitation, wind, sound, and lights in a simulated thunderstorm. A greenhouse, livestock barn, or other setting for growing living entities can synchronize ambient lighting with automated feeding and watering devices. Any combination of electromechanical devices can be timed and/or coordinated by the systems and methods described herein. Such devices may be represented on an interface for creating the

sequence as additional lines on a grid, e.g., one line for each separate component being controlled, or by any other suitable means. Effects of these other devices can also be visually represented to the user. For instance, continued use of a smoke machine could slowly haze out other grids, a coffee maker could be represented by a small representation of a coffee maker that appears to brew coffee on the interface as the action occurs at the device or the interface can show a bar slowing changing color as feed is dispensed in a livestock barn. Other such static or dynamic effects would be readily apparent to one of skill in the art and are all incorporated within this disclosure.

In certain embodiments, wherein the lighting units are capable of motion, e.g., by sliding, pivoting, rotating, tilting, etc., the user may include instructions for the motion or movement of lighting units. This function may be accomplished by any means. For example, if the lighting unit includes a motor or other system capable of causing movement, the desired movement may be effected by selecting a motion effect from a set of motion effects, as described for lighting effects above. Thus, for example, a lighting unit capable of rotating on its base may be selected, and a rainbow wash effect may be programmed to occur simultaneously with a rotating motion effect. In other embodiments, lighting units may be mounted on movable platforms or supports which can be controlled independently of the lights, e.g., by providing an additional line on a grid interface as described above. Motion effects may also have parameters, such as speed and amount (e.g., an angle, a distance, etc.), that can be specified by the user. Such light/motion combinations may be useful in a wide variety of situations, such as light shows, planetarium presentations, moving spotlights, and any other scenario in which programmable moving lights may be desirable.

Similarly, instructions for controlling objects placed between a lighting unit and an object being illuminated, such as gobos, stencils, filters, lenses, irises and other objects through which light may pass, can be provided by a user according to the systems and methods described herein. In this manner, an even wider array of lighting effects may be designed and preprogrammed for later execution.

One embodiment of the systems and methods described herein is a computer system, such as processor 10 depicted in Figure 1, configured to design or create a lighting sequence according to the systems and methods described herein, e.g., by executing a computer program in a computer language either interpreted or compiled,

e.g., Fortran, C, Java, C++, etc. In an additional embodiment, the systems and methods described herein relate to a disk, CD, or other permanent computer-readable storage medium that encodes a computer program capable of performing some or all of the functions described above which enable a user to create or design a lighting sequence which can be used to control a plurality of lighting units.

A lighting sequence may be recorded on a storage medium, such as a compact disk, floppy disk, hard drive, magnetic tape, volatile or non-volatile solid state memory device, or any other permanent computer-readable storage medium. The lighting sequence may be stored in a manner that records the effects and their parameters as created by a user, in a manner that converts that format into a format which represents the final data stream, e.g., suitable for directly controlling lighting units or other devices, or in any other format suitable for executing the lighting sequence. In embodiments wherein the sequence is stored as a data stream, the system may permit a user to choose from a selection of data formats such as DMX, RS-485, RS-232, etc. Additionally, lighting sequences may be linked to each other, e.g., such that at the conclusion of one sequence, another sequence is executed, or a master sequence may be created for coordinating the execution of a plurality of subsequences, e.g., based on external signals, conditions, time, randomly, etc. In certain embodiments, a lighting sequence 20 may be executed directly from a processor 10, although in other embodiments, a lighting sequence 20 may be executed using a controller 30 as described below.

A controller 30, as depicted in Figure 6, may be used to execute lighting sequences 20 which have been programmed, designed, or created on a different apparatus. Because the controller 30 may provide a narrower range of functions than the processor used to create the sequence, the controller 30 may contain less hardware and be less expensive than a more complex system which permits authoring, includes a video monitor, or has other auxiliary functionality. The controller 30 may employ any suitable loader interface 610 for receiving a lighting program 20, e.g., an interface for reading a lighting program 20 from a storage medium such as a compact disk, diskette, magnetic tape, smart card, or other device, or an interface for receiving a transmission from another system, such as a serial port, USB port, parallel port, IR receiver, or other connection for receiving a lighting program 20. In certain embodiments, the lighting

program 20 may be transmitted over the Internet. The controller 30 may also include an interface for communicating with a plurality of lighting units 40.

A controller 30 may begin execution of a lighting sequence 20 upon loading the lighting sequence 20, upon receiving a command or signal from a user or a device or sensor, at a specified time, or upon any other suitable condition. The condition for initiation may be included in the lighting sequence 20, or may be determined by the configuration of the controller 30. Additionally, in certain embodiments, the controller may begin execution of a lighting sequence 20 starting from a point in the middle of the lighting sequence 20. For example, the controller 30 may, upon receiving a request from the user, execute a lighting sequence 20 starting from a point three minutes from the beginning of the sequence, or at any other specified point, e.g., from the fifth effect, etc. The controller 30 may, upon receiving a signal from a user or a device or sensor, pause the playback, and, upon receiving a suitable signal, resume playback from the point of pausing. The controller may continue to execute the lighting sequence 20 until the sequence terminates, until a command or signal is received from a user or a device or sensor, until a specified time, or until any other suitable condition.

A controller 30 may include a memory unit, database, or other suitable module 620 for storing a plurality of predetermined stock effects and instructions for converting those effects into a data format, such as DMX, RS-485, or RS-232, suitable for controlling a plurality of lighting units. The memory module 620 may be preconfigured for a set of stock effects, the memory module 620 may receive effects and instructions from the lighting sequence 20, or the memory module 620 may include a preconfigured set of stock effects which can be supplemented by additional effects stored in lighting sequence 20. Preconfiguring the memory module 620 with a set of stock effects permits a reduction in the memory required to store a lighting sequence 20, because the lighting sequence 20 may omit conversion instructions for effects preconfigured into the controller 30. In embodiments wherein the lighting sequence 20 includes stock effects designed by the author, suitable instructions may be included in lighting sequence 20 and stored in memory module 620, e.g., upon loading or execution of the lighting sequence 20.

The controller 30 may include an external interface 650 whereby the controller 30 can receive external signals useful for modifying the execution of the lighting

sequence 20. For example, the external interface 650 may include a user interface, which may in turn include switches, buttons, dials, sliders, a console, a keyboard, or any other device, such as a sensor, whereby a user may provide a command or signal to the controller 30 or otherwise influence the execution or output of the lighting sequence 20. The external interface 650 may receive temporal information from one or more chronometers, such as a local time module 660 which functions as a counter for measuring time from a predetermined starting point, such as when the controller 30 is turned on or when the counter is reset, or a date time module 665 which calculates the current date and time. Additionally, the controller 30 may receive commands or signals from one or more external devices or sensors through external input 668. Such devices may be coupled to controller 30 directly, or signals may be received by the controller through an IR sensor or other suitable interface. Signals received by the controller 30 may be compared to or interpreted by a cue table 630, which may contain information relating to the various inputs or conditions designated by the author of the lighting sequence 20 to affect the execution or output of the lighting sequence 20. Thus, if the controller 30 compares an input to the cue table 630 and determines that a condition has been satisfied or a designated signal has been received, the controller 30 may then alter the execution or output of the lighting sequence 20 as indicated by the program.

In certain embodiments, the controller may respond to external signals in ways that are not determined by the contents and instructions of the lighting sequence 20. For example, the external interface 650 may include a dial, slider, or other feature by which a user may alter the rate of progression of the lighting sequence 20, e.g., by changing the speed of the local time counter 660, or by altering the interpretation of this counter by the controller 30. Similarly, the external interface 650 may include a feature by which a user may adjust the brightness, color, or other characteristic of the output. In certain embodiments, a lighting sequence 20 may include instructions to receive a parameter for an effect from a feature or other user interface on the external interface 650, permitting user control over specific effects during playback, rather than over the output or system of lighting units as a whole.

The controller 30 may also include a transient memory 640. The transient memory 640 may store temporary information, such as the current state of each lighting unit under its control, which may be useful as a reference for the execution of the

lighting sequence 20. For example, as described above, some effects may use output of another effect to define a parameter; such effects may retrieve the output of the other effect as it is stored in the transient memory 640. Those of skill in the art will recognize other situations in which a transient memory 640 may be useful, and such uses are intended to be encompassed by the present disclosure.

The controller 30 may send the data created by the execution of lighting sequence 20 to lighting units by providing the data to a network output 680, optionally through the intermediacy of an output buffer 670. Signals to additional devices may be transmitted through the network out put 680, or through a separate external output 662, as convenient or desirable. The data may be transmitted through data connections such as wires or cables, as IR or RF transmissions, other suitable methods for data transfer, or any combination of methods capable of controlling lighting units and/or other devices.

In certain embodiments, the controller 30 may not communicate directly with the lighting units, but may instead communicate with one or more subcontrollers which, in turn, control the lighting units or another level of subcontrollers, etc. The use of subcontrollers permits distributive allocation of computational requirements. An example of such a system which uses this sort of distributional scheme is disclosed in U.S. Patent No. 5,769,527 to Taylor, described therein as a "master/slave" control system. For the systems and methods described herein, communication between the various levels may be unidirectional, wherein the controller 30 provides instructions or subroutines to be executed by the subcontrollers, or bidirectional, where subcontrollers relay information back to the controller 30, for example, to provide information useful for effects which rely on the output of other effects as described above, for synchronization, or for any other conceivable purpose.

Although the description above illustrates one particular configuration of a controller 30, other configurations for achieving the same or similar functions will be apparent to those of skill in the art, and such variations and modifications are intended to be encompassed by the present invention. The example below more particularly describes an embodiment of a controller 30 such as described above.

The following describes one embodiment of a controller according to the systems and methods described herein, as exemplified in Figure 6, including the design and

format of a show representation, management of external inputs and outputs, interpretation and execution of shows, and generation of DMX compliant output. The controller architecture of this embodiment uses a Java-based object-oriented design; however, other object-oriented, structured, or other programming languages may be used with the invention.

The controller architecture permits effects to be based on external environmental conditions or other input. An effect is a predetermined output involving one or more lighting units. For example, fixed color, color wash, and rainbow wash are all types of effects. An effect may be further defined by one or more *parameters,* which specify, for example, lights to control, colors to use, speed of the effect, or other aspects of an effect. The *environment* refers to any external information that may be used as an input to modify or control an effect, such as the current time or external inputs such as switches, buttons, or other transducers capable of generating control signals, or events generated by other software or effects. Finally, an effect may contain one or more *states*, so that the effect can retain information over the course of time. A combination of the state, the environment, and the parameters may be used to fully define the output of an effect at any moment in time, and over the passage of time

In addition, the controller may implement effect priorities. For example, different effects may be assigned to the same lights. By utilizing a priority scheme, only the highest priority effect will determine the light output. When multiple effects control a light at the same priority the final output may be an average or other combination of the effect outputs.

A lighting sequence as described above may be deployed as a program fragment. Such fragments may be compiled in an intermediate format, such as by using an available Java compiler to compile the program as byte codes. In such a byte code format, the fragment may be called a sequence. A sequence may be interpreted or executed by the controller 30. The sequence is not a stand-alone program, and adheres to a defined format, such as an instantiation of an object from a class, that the controller 30 may use to generate effects. When downloaded into the controller 30 (via serial port, infrared port, smart card, or some other interface), the controller 30 interprets the sequence, executing portions based on time or input stimuli.

A building block for producing a show is an effect object. The effect object includes instructions for producing one specific effect, such as color wash, cross fade, or fixed color, based on initial *parameters* (such as which lights to control, start color, wash period, etc.) and inputs (such as time, environmental conditions, or results from other effect objects). The sequence contains all of the information to generate every effect object for the show. The controller 30 instantiates all of the effect objects one time when the show is started, then periodically sequentially activates each one. Based on the state of the entire system, each effect object can programmatically decide if and how to change the lights it is controlling.

The run-time environment software running on the controller 30 may be referred to as a conductor. The conductor may be responsible for downloading sequences, building and maintaining a list of effect object instances, managing the interface to external inputs and outputs (including DMX), managing the time clock, and periodically invoking each effect object. The conductor also maintains a memory that objects can use to communicate with each other.

The controller 30 may maintain two different, but synchronized, representations of time. The first is LocalTime, which is the number of milliseconds since the controller 30 has been turned on. LocalTime may be represented as a 32-bit integer that will roll over after reaching its maximum value. The other time representation is DateTime, which is a defined structure maintaining the time of day (to seconds resolution) as well as the day, month, and year.

LocalTime may be used by effects for computing relative changes, such as a hue change since last execution in a color wash effect. LocalTime roll-over should not cause effects to fail or malfunction. The conductor may provide utility functions for common operations like time deltas.

An effect object may be an instance of an Effect class. Each effect object may provide two public methods which are subclassed from Effect to produce the desired effect. These are the constructor and the run() methods.

The constructor method may be called by a sequence when an instance of the effect is created. It can have any number and type of parameters necessary to produce the

desired effect variations. The authoring software may be responsible for producing the proper constructor parameters when creating the sequence.

The first argument to the constructor may be an integer identifier (ID). The ID may be assigned by the show authoring software, and may be unique.

The constructor may call super() to perform any conductor-specific initializations.

The effect class may also contain next and prev members, which are used by the sequence and conductor to maintain a linked list of effects. These members may not be accessed internally by the effect methods.

Certain typical effects may be used over and over again. These typical effects may be provided by the conductor, minimizing the storage/download size of sequences. The typical effects may, if desired, be further sub-classed.

A sequence is a convenient means of bundling together all of the information necessary to produce a show. The sequence may have only one required public method, init(), which is called once by the conductor prior to running the show. The init() method may instantiate every effect used by the show, passing the ID and any parameters as constructor arguments. The init() method may then link the effect objects together into a linked list, and return the list to the conductor.

The linked list is maintained through the next and prev members of the effect objects. The prev member of the first object is nil, and the next member of the last object is nil. The first effect is returned as the value of init().

The optional dispose() method will be called when the sequence is deactivated. This method can be used to clean up any resources allocated by the sequence. Automatic processes may be used independently to handle any allocated memory. The base class dispose() will pass through the linked list and free the effect objects, so when dispose() is subclassed, it may be necessary to call super().

The optional public method String getSequenceInfo() can be used to return version and copyright information. It may be desirable to implement some additional

getSequence*() routines to return information that may be useful for the controller/user interface.

A sequence may require additional supporting classes. These may be included, along with the sequence object, in a file such as a JAR (Java ARchive) file. The JAR file may then be downloaded to the conductor. Tools for JAR files are part of the standard Java development tools.

Any DMX communication may be handled by a DMX_Interface class. Each instance of a DMX_Interface controls one DMX universe. The DMX_Interface base class may be sub-classed to communicate over a specific type of hardware interface (serial, parallel, USB).

A *channel* may be a single data byte at a particular location in the DMX universe. A *frame* may be all of the channels in the universe. The number of channels in the universe is specified when the class is instantiated.

Internally, DMX_Interface maintains three buffers, each the length of the number of channels: the last frame of channels that was sent, the next frame of channels waiting to be sent, and the most recent priority of the data for each channel. Effect modules may modify the channel data waiting to be sent via the SetChannel() method, and the conductor may ask for the frame to be sent via SendFrame().

When an effect object sets the data for a particular channel it may also assign that data a priority. If the priority is greater than the priority of the last data set for that channel, then the new data may supercede the old data. If the priority is lesser, then the old value may be retained. If the priorities are equal, then the new data value may be added to a running total and a counter for that channel may be incremented. When the frame is sent, the sum of the data values for each channel may be divided by the channel counter to produce an average value for the highest priority data.

After each frame has been sent, the channel priorities may all be reset to zero. The to-be-sent data may be retained, so if no new data is written for a given channel it will maintain its last value, and also copied to a buffer in case any effect objects are interested.

An exemplary DMX_Interface may implement the following methods:

A DMX_Interface(int num_channels) method is a constructor that sets up a DMX universe of num_channels (24 .. 512) channels. When subclassed, the method may take additional arguments to specify hardware port information.

A void SetChannel(int channel, int data, int priority) method sets the to-be-sent data (0 .. 255) for the channel if the priority is greater than the current data priority. The method can throw error handling exceptions, such as ChannelOutOfRange and DataOutOfRange exceptions.

A void SetChannels(int first_channel, int num_channels, int data[], int priority) method sets num_channels of to-be-sent data for starting with first_channel from the array data. The method can throw error handling exceptions, such as ChannelOutOfRange, DataOutOfRange, and ArrayIndexOutOfBounds exceptions.

A int GetChannelLast(int channel) method returns the last data sent for the channel. The method can throw error handling exceptions, such as ChannelOutOfRange or NoDataSent exceptions.

A void SendFrame(void) method causes the current frame to be sent. This is accomplished through a separate thread so processing by the conductor will not pause. If a frame is already in progress, it is terminated and the new frame started.

A int FrameInProgress(void), if no frame is currently being sent, returns zero. If a frame is in progress, it returns the number of the last channel sent.

The conductor is the run-time component of the controller that unites the various data and input elements. The conductor may download sequences, manage the user interface, manage the time clock and other external inputs, and sequence through the active effect objects.

The technique for downloading the sequence JAR file into the conductor can vary depending on the hardware and transport mechanism. Various Java tools can be utilized for interpreting the JAR format. In an embodiment, the sequence object and various required classes may be loaded into memory, along with a reference to the sequence object.

In an embodiment, more than one sequence object may be loaded into the conductor, and only one sequence may be active. The conductor can activate a sequence based on external inputs, such as the user interface or the time of day.

If a sequence is already active, then prior to activating a new sequence, the dispose() method is invoked for the already active sequence.

To activate a sequence, the sequence's init() method is called and run to completion.

Controllers may invoke some method for measuring time. Time values may be accessed via GetLocalTime() and GetDateTime() methods. Other inputs may be enumerated and accessed by a reference integer. The values of all inputs may also be mapped to integers. A GetInput(int ref) method returns the value of input ref, and can throw exceptions, such as a NoSuchInput exception.

The effect list may be created and returned by the sequence's init() method. At fixed intervals the conductor may sequentially call the run() method of each effect object in the list.

The interval may be specific to the particular controller hardware, and may be alterable, e.g., by an external interface. If the effect list execution does not finish in one interval period, the next iteration may be delayed until the following interval time. Effect objects may not need to run every interval to compute changes, but may use a difference between the current time and the previous time.

Effects may be designed to minimize the use of processing power, so the entire effect list can be run quickly. If an effect requires a large amount of computation, it may initiate a low priority thread to do the task. While the thread is running, the run() method may return right away, so the lights will remain unchanged. When the run() method detects that the thread has finished, it may use the results to update the light outputs.

The memory allows different effects to communicate with each other. Like external inputs, memory elements may be integers. Memory elements may be referenced by two pieces of information: the ID of the effect that created the information, and a reference integer which is unique to that effect. The accessor methods are:

void SetScratch(int effect_id, int ref_num, int value)

int GetScratch(int effect_id, int ref_num)

Both methods can throw error handling exceptions, such as NoSuchEffect and NoSuchReference exceptions.

Effects may run in any order. Effects that use results from other effects may anticipate receiving results from the previous iteration.

Additional routines may include the following.

An int DeltaTime(int last) method computes the change in time between the current time and last.

A DMX_Interface GetUniverse(int num) method returns the DMX_Interface object associated with universe number num. This value should not change while a sequence is running, so it can be cached. The method can throw error handling exceptions, such as NoSuchUniverse exceptions.

An int[] HSBtoRGB(int hue, int sat, int bright) method converts hue (0 – 1535), saturation (0 – 255), and brightness (0 – 255) in to red/green/blue values, which are written to the first three elements of the resulting array. The method can throw error handling exceptions, such as ValueOutOfRange exceptions.

An int LightToDMX(int light) method returns the DMX address of a light with a logical number of light. The method can throw error handling exceptions, such as DMXAddressOutOfRange exceptions.

A void LinkEffects(Effect a, Effect b) method sets a.next = b; b.prev = a.

Each controller may have a configuration file used by the show authoring software. The configuration file may contain mappings between the input reference integers and more useful descriptions of their functions and values, for example, something like: Input 2 = "Slider" range = (0 – 99). The configuration file can also contain other useful information, such as a number of DMX universes.

The following is an example of code illustrating a lighting sequence authored according to the principles of the invention. It will be understood that the following example is in no way limiting:

Example 1:

```
// An example sequence.
// Runs one strip of 12 cove lights, sequentially numbered starting at address 1
// Input #1 is a binary switch
// The cove runs a continuous color wash
// When the switch is opened, a chaser strobe effect is triggered, which runs a
// white strobe down cove. The effect won't be repeated until the switch is reset.

import java.sequence.*

public class ExampleSequence extends Sequence {
  private int CoveGroup[] = {
    LightToDMX(1), LightToDMX(2), LightToDMX(3), LightToDMX(4),
    LightToDMX(5), LightToDMX(6), LightToDMX(7), LightToDMX(8),
    LightToDMX(9), LightToDMX(10), LightToDMX(11), LightToDMX(12)
  };

  public String getSequenceInfo() {
    return "Example sequence version 1.0";
  }

  public Effect init() {
    super.init();                          // Call base class init

    // Create the effect objects with the appropriate variation params
    washEff = new WashEffect(
      1,                                   // ID
      CoveGroup, 1, 1,                     // Which lights, universe 1, priority 1
      true,                                // Direction = forward
```

```
        20000);                         // Speed (20 seconds)

      strobeEff = new ChaseStrobeEffect(
        2,                              // ID
        CoveGroup, 1, 2,                // Which lights, universe 1, priority 2
        1,                              // Trigger input
        true,                           // Direction = forward
        100,                            // Duration of strobe (100 ms)
        400,                            // Time between strobes (400 ms)
        255, 255, 255);                 // Strobe color (white)

      // Link the effects
      LinkEffects(washEff, strobeEff);  // Sets next and prev

      // Return the effect list to the conductor
      return(washEff);
    }

    // Declare all of the effects
    WashEffect washEff;
    ChaseStrobeEffect strobeEff;
}

// WashEffect may be implemented as a stock effect, but we'll implement a simple
// version here for illustration.

public class WashEffect extends Effect {
  private int hue, sat, bright;
  private int last_time;                // Last time we ran
  private int lights[];
  private DMX_Interface universe;
  private int priority;
  private boolean direction;
```

```
    private int speed;

    public WashEffect(int id, int lights[], int univ, int prio, boolean dir, int speed) {
      // make copies of variation params and initialize any
      // other variables
      this.lights = lights;
      this.universe = GetUniverse(univ);
      this.priority = prio;
      this.direction = dir;
      this.speed = speed;
      hue = 0;
      sat = 255;
      bright = 255;
      last_time = 0;
      super(id);
    }
```

All articles, patents, and other references set forth above are hereby incorporated by reference. While the invention has been disclosed in connection with the embodiments shown and described in detail, various equivalents, modifications, and improvements will be apparent to one of ordinary skill in the art from the above description. Such equivalents, modifications, and improvements are intended to be encompassed by the following claims.